1 数据集视频准备

2024年8月10日 16:28

命令行

cd ava_made/Dataset

bash cut_video.sh

bash cut_frames.sh

python choose_frames_all.py 46 0

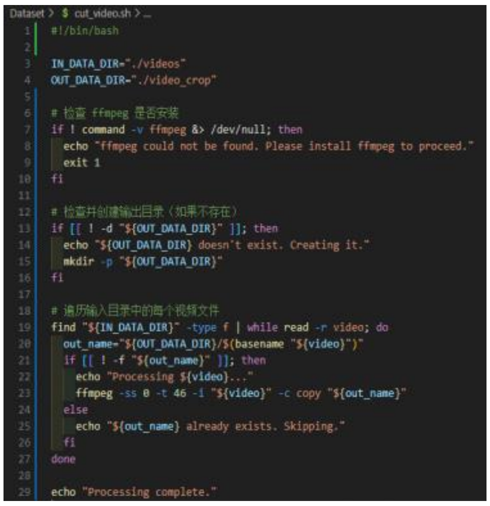
python choose_frames.py 46 0

1 数据集视频准备

使用船上视频，截取片段，首先对视频裁剪与抽帧，使用ffmpeg进行视频裁剪与抽帧

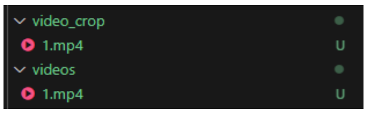
1.1视频裁剪 指定输入和输出文件夹，将文件夹内的视频裁剪成全部一致的长度，

如：ffmpeg -ss 0 -t 46 -i "${video}" -c copy "${out_name}" 就是截取视频的0-46秒的部分



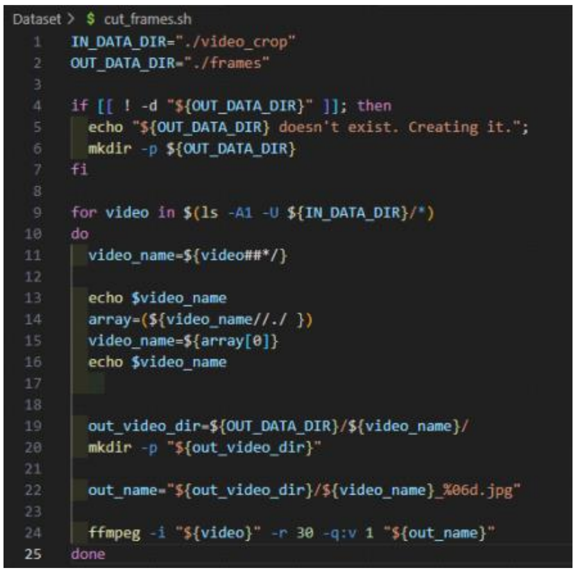
Dataset > $ cut_video.sh > …

```
#!/bin/bash

IN_DATA_DIR="./videos"
OUT_DATA_DIR="./video_crop"

# 检查 ffmpeg 是否安装
if ! command -v ffmpeg &> /dev/null; then
  echo "ffmpeg could not be found. Please install ffmpeg to proceed."
  exit 1
fi

# 检查并创建输出目录（如果不存在）
if [[ ! -d "${OUT_DATA_DIR}" ]]; then
  echo "${OUT_DATA_DIR} doesn't exist. Creating it."
  mkdir -p "${OUT_DATA_DIR}"
fi

# 遍历输入目录中的每个视频文件
find "${IN_DATA_DIR}" -type f | while read -r video; do
  out_name="${OUT_DATA_DIR}/$(basename "${video}")"
  if [[ ! -f "${out_name}" ]]; then
    echo "Processing ${video}..."
    ffmpeg -ss 0 -t 46 -i "${video}" -c copy "${out_name}"
  else
    echo "${out_name} already exists. Skipping."
  fi
done

echo "Processing complete."
```



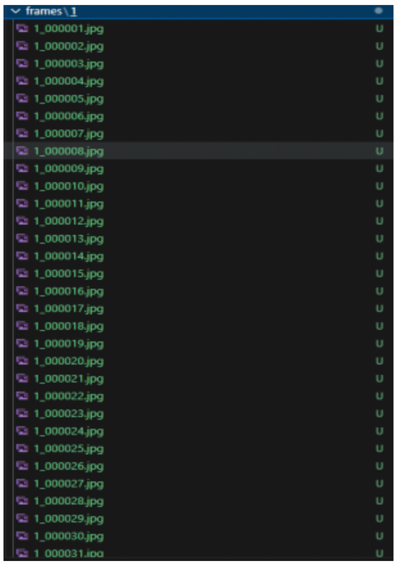
```
video_crop
  1.mp4      U
videos
  1.mp4      U
```

1.2视频抽帧 将上一步裁剪后的视频抽帧，参考ava数据集，每秒裁剪30帧 如： ffmpeg -i "${video}" -r 30 -q:v 1 "${out_name}" 即将一秒分为30帧，一秒有30张图片



Dataset > $ cut_frames.sh

```
IN_DATA_DIR="./video_crop"
OUT_DATA_DIR="./frames"

if [[ ! -d "${OUT_DATA_DIR}" ]]; then
  echo "${OUT_DATA_DIR} doesn't exist. Creating it.";
  mkdir -p ${OUT_DATA_DIR}
fi

for video in $(ls -A1 -U ${IN_DATA_DIR}/*)
do
  video_name=${video##*/}

  echo $video_name
  array=(${video_name//./ })
  video_name=${array[0]}
  echo $video_name


  out_video_dir=${OUT_DATA_DIR}/${video_name}/
  mkdir -p "${out_video_dir}"

  out_name="${out_video_dir}/${video_name}_%06d.jpg"

  ffmpeg -i "${video}" -r 30 -q:v 1 "${out_name}"
done
```



```
frames\1
  1_000001.jpg   U
  1_000002.jpg   U
  1_000003.jpg   U
  1_000004.jpg   U
  1_000005.jpg   U
  1_000006.jpg   U
  1_000007.jpg   U
  1_000008.jpg   U
  1_000009.jpg   U
  1_000010.jpg   U
  1_000011.jpg   U
  1_000012.jpg   U
  1_000013.jpg   U
  1_000014.jpg   U
  1_000015.jpg   U
  1_000016.jpg   U
  1_000017.jpg   U
  1_000018.jpg   U
  1_000019.jpg   U
  1_000020.jpg   U
  1_000021.jpg   U
  1_000022.jpg   U
  1_000023.jpg   U
  1_000024.jpg   U
  1_000025.jpg   U
  1_000026.jpg   U
  1_000027.jpg   U
  1_000028.jpg   U
  1_000029.jpg   U
  1_000030.jpg   U
  1_000031.jpg   U
```

1.3 整合与缩减帧

1.2节中产生的frames文件夹的结构，（现在仅用了一个视频做试验）在后续yolo检测时会出现不方便，将所有的图片放在了一个文件夹（choose_frames_all）中。

同时，并不是，所有图片都需要检测与标注，假如在10秒的视频中，检测标注：x_000001.jpg、x_000031.jpg、x_000061.jpg、x_000091.jpg、x_0000121jpg、x_000151.jpg、x_000181.jpg、x_000211.jpg、x_000241.jpg、x_000271.jpg、x_000301.jpg。

1.4 不整合的缩减

1.3的整合与缩减是为了yolov10的检测，这里的不整合的缩减是为了via的标注。

2 YOLO

2024年8月10日　　17:03

命令行

cd ava_made/yolovdeepsort/v10

python train.py

python detect2.py

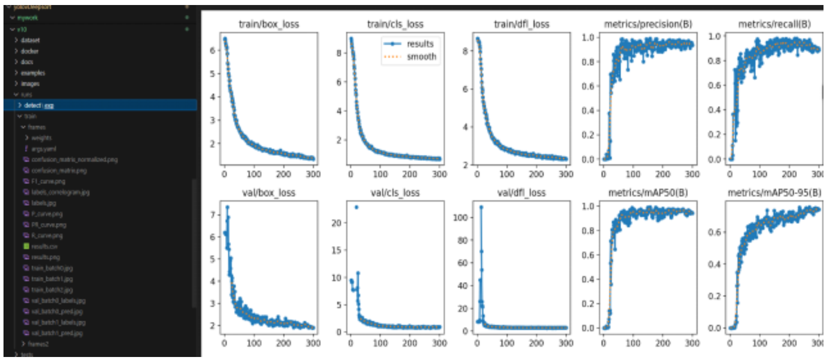
2.1安装YOLOv10

2.2 训练YOLO检测的权重（也可使用官方预训练权重yolov10n/s/l.pt）

创建一个将来要检测的视频的数据集用于YOLOv10的训练（在第一步中已经将视频抽帧完毕，选取一定量的图片创建一个数据集进行标注即可）,训练结果如下

权重文件保存在：**D:\github_files\slowfast-mmaction-ava\ava_made\yolovdeepsort\v10\runs\train\frames\weights\best.pt**

```
import warnings
warnings.filterwarnings('ignore')
from ultralytics import YOLO

if __name__ == '__main__':
    model = YOLO('./yolovDeepsort/v10/ultralytics/cfg/models/v10/yolov10n.yaml')
    model.train(data='./yolovDeepsort/v10/dataset/frames/frmes.yaml',
                cache=False,
                imgsz=640,
                epochs=300,
                batch=32,
                close_mosaic=0,
                workers=4,
                # device='0',
                optimizer='SGD', # using SGD
                patience=0, # close earlystop
                # resume=True, # 断点续训,YOLO初始化时选择last.pt
                # amp=False, # close amp
                # fraction=0.2,
                project='runs/train',
                name='frames',
                )
```

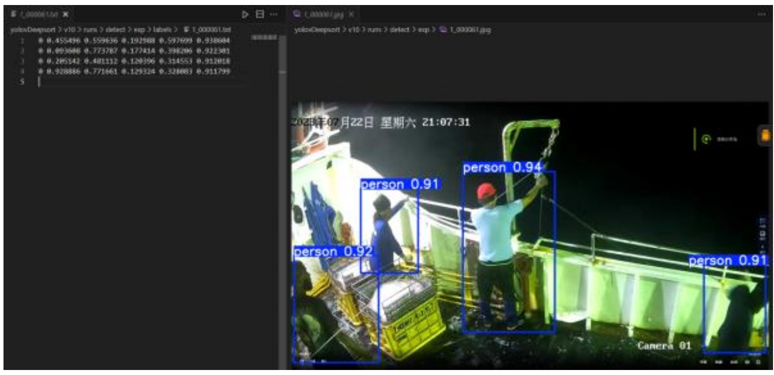


2.3 使用YOLOv10对choose_frames_all进行检测

结果保存在：**D:\github_files\slowfast-mmaction-ava\ava_made\yolovdeepsort\v10\runs\detect\exp**

```
import warnings
warnings.filterwarnings('ignore')
from ultralytics import YOLO
import os

def save_results_with_confidence(results, save_dir):
    for result in results:
        # 获取图片文件名并构造对应的txt文件名
        file_name = os.path.splitext(os.path.basename(result.path))[0]
        txt_path = os.path.join(save_dir, f"{file_name}.txt")
        with open(txt_path, 'w') as f:
            for box in result.boxes:
                # 提取检测框的信息
                class_id = int(box.cls)
                confidence = box.conf[0]
                x_min, y_min, x_max, y_max = box.xyxy[0]

                # 计算中心坐标和宽高并归一化
                img_width = result.orig_shape[1]
                img_height = result.orig_shape[0]
                x_center = (x_min + x_max) / 2 / img_width
                y_center = (y_min + y_max) / 2 / img_height
                width = (x_max - x_min) / img_width
                height = (y_max - y_min) / img_height

                # 将类别ID、中心坐标、宽高和置信度写入txt文件
                f.write(f"{class_id} {x_center:.6f} {y_center:.6f} {width:.6f} {height:.6f} {confidence:.6f}\n")
if __name__ == '__main__':
    model = YOLO('./yolovDeepsort/v10/runs/train/frames/weights/best.pt') # select your model.pt path
    results = model.predict(
        source='dataset/choose_frames_all',
        imgsz=640,
        project='runs/detect',
        name='exp',
        save=True,
        save_txt=False,  # 不使用内置的save_txt, 手动保存
    )
    # 指定保存路径
    save_dir = os.path.join('runs/detect', 'exp', 'labels')
    os.makedirs(save_dir, exist_ok=True)
    # 保存带置信度的检测结果
    save_results_with_confidence(results, save_dir)
```

得到.txt文件，包括对应图片中检测出来的人的锚框的位置，以及置信度大小



```
yolovDeepsort > v10 > runs > detect > exp > labels > 1_000001.txt
0 0.455496 0.559636 0.192988 0.597699 0.938604
0 0.093608 0.773787 0.177414 0.398296 0.922301
0 0.205142 0.481112 0.120396 0.314553 0.912018
0 0.928886 0.771661 0.129324 0.328003 0.911799
```

yolovDeepsort > v10 > runs > detect > exp > 1_000001.jpg
2023年07月22日 星期六 21:07:31
person 0.91
person 0.94
person 0.92
person 0.91
Camera 01

3 via标注

2024年8月10日 17:37

3.1生成dense_proposals_train.pkl，目的是为了导入via做准备

根据第二步使用YOLOv10检测出来的标签.txt文件生成.pkl文件

```
import sys
import os
import json
import pickle
#传参 labelPath是yolov10检测结果的位置，需要获取0 (0代表人) 的四个坐标值，
还需要检测概率
# ../yolov10/runs/detect/exp/labels
labelPath = sys.argv[1]
#传参 保存为pkl的地址，这是像ava数据集对齐
# ./avaMin_dense_proposals_train.pkl
avaMin_dense_proposals_path = sys.argv[2]
#传参 是否可视化
#可见就传入 show，否则不填
showPkl = sys.argv[3]
results_dict = {}
dicts = []
for root, dirs, files in os.walk(labelPath):
    lenFile = len(files)
    if root == labelPath:
        # 排序，防止10排在1的前面
        files.sort(key=lambda arr: (int(arr[:-7]), int(arr[3:-4])))

        # 处理文件的范围：跳过前两个和后两个
        for idx, name in enumerate(files):
            if idx < 2 or idx >= lenFile - 2:
                continue

            temp_file_name = name.split("_")[0]
            temp_video_ID = name.split("_")[1].split('.')[0]
            temp_video_ID = int(temp_video_ID)
            temp_video_ID = str(int((temp_video_ID-1)/30))
            temp_video_ID = temp_video_ID.zfill(4)
            # key = '视频名字,第几秒的视频' 如 '1,0002', 代表视频1的第2秒
            key = temp_file_name + ',' + temp_video_ID
            # 读取yolov10中的信息
            temp_txt = open(os.path.join(root, name))
            temp_data_txt = temp_txt.readlines()
            results = []
            for i in temp_data_txt:
                # 只要人的信息
                j = i.split(' ')
                if j[0] == '0':
                    # 由于yolo10的检测结果是 xywh
                    # 要将xywh转化成xyxy
                    y = j
                    y[1] = float(j[1]) - float(j[3]) / 2  # top left x
                    y[2] = float(j[2]) - float(j[4]) / 2  # top left y
                    y[3] = float(j[1]) + float(j[3])  # bottom right x
                    y[4] = float(j[2]) + float(j[4])  # bottom right y
                    results.append([y[1], y[2], y[3], y[4], y[5]])
                    dicts.append([y[1], y[2], y[3], y[4], y[5]])
            temp_txt.close()  # 关闭文件
            results_dict[key] = results

# 保存为pkl文件
with open(avaMin_dense_proposals_path,"wb") as pklfile:
    pickle.dump(results_dict, pklfile)

# 显示pkl文件中的内容
if showPkl == "show":
    for i in results_dict:
        print(i,results_dict[i])
        #input()
```

命令行

```
cd  ava_made/Dataset/yolovDeepsort/mywork

python dense_proposals_train.py ../v10/runs/detect/exp/labels ./dense_proposals_train.pkl show

cd  ava_made/Dataset

python choose_frames_middle.py

cd  ava_made/Dataset/yolovDeepsort/mywork

python dense_proposals_train_to_via.py ./dense_proposals_train.pkl ../../Dataset/choose_frames_middle/

cd  ava_made/Dataset

python chang_via_json.py
```

3.2 choose_frames_all_middle

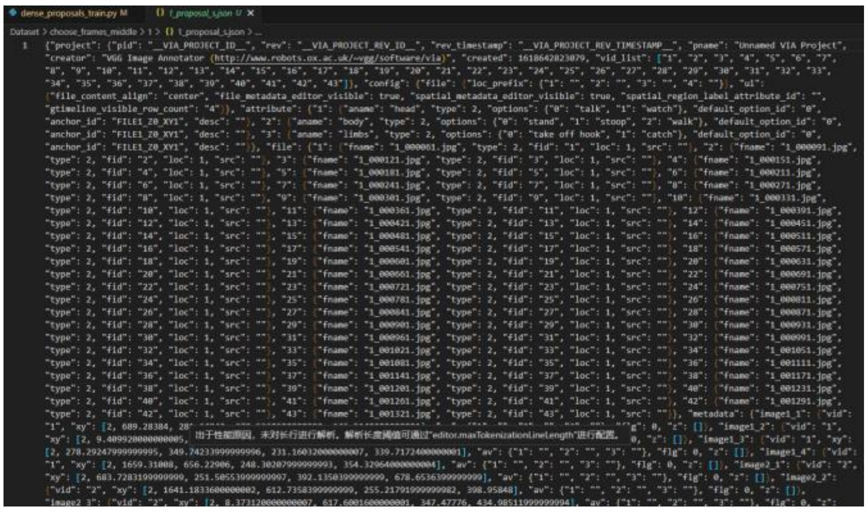
Dataset 下的 choose_frames 文件夹中的包含46秒的47张图片，但是在最后生成的标注文件，不包含前2张图片和后2张图片。所以需要创建一个choose_frames_middle文件夹，存放不含前2张图片与后2张图片的文件夹。

3.3生成via标注文件（.json文件）

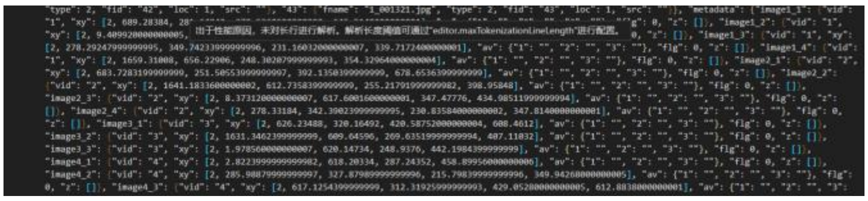
生成的标注文件保存在：Dataset/choose_frames_middle\1\1.proposal.json

自定义动作名称如：
头部：说话，看
身体：站立，弯腰，走动
四肢：摘钩，抓

```
from via3_tool import Via3Json
import pickle
import csv
from collections import defaultdict
import os
import cv2
import sys
#传参 ./avaMin_dense_proposals_train.pkl
avaMin_dense_proposals_path = sys.argv[1]
#传参 ../videoData/choose_frames/
json_path = sys.argv[2]

f = open(avaMin_dense_proposals_path,'rb')
info = pickle.load(f, encoding='iso-8859-1')



attributes_dict = {'1':dict(aname='head', type=2, options={'0':'talk',
                  '1':'watch'},default_option_id="", anchor_id = 'FILE1_Z0_XY1'),
                  '2': dict(aname='body', type=2, options={'0':'stand',
                  '1':'stoop', '2':'walk'}, default_option_id="", anchor_id='FILE1_Z0_XY1'),

                  '3':dict(aname='limbs', type=2, options={'0':'take off hook',
                  '1':'catch'},default_option_id="", anchor_id = 'FILE1_Z0_XY1'),
                  }
#len_x与循环的作用主要是获取每个视频下视频帧的数量
dirname = ''
len_x = {}
for i in info:
    temp_dirname = i.split(' ')[0]
```


dense_proposals_train.py M
1_proposal_s.json U
Dataset > choose_frames_middle > 1 > {} 1_proposal_s.json > ...
由于性能原因，未对长行进行解析。解析长度阈值可通过"editor.maxTokenizationLineLength"进行配置。

```
{"project": {"pid": "__VIA_PROJECT_ID__", "rev": "__VIA_PROJECT_REV_ID__", "rev_timestamp": "__VIA_PROJECT_REV_TIMESTAMP__", "pname": "Unnamed VIA Project",
"creator": "VGG Image Annotator (http://www.robots.ox.ac.uk/~vgg/software/via)", "created": 1618642823079, "vid_list": ["1", "2", "3", "4", "5", "6", "7",
"8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33",
"34", "35", "36", "37", "38", "39", "40", "41", "42", "43"]}, "config": {"file": {"loc_prefix": {"1": "", "2": "", "3": "", "4": ""}}, "ui":
{"file_content_align": "center", "file_metadata_editor_visible": true, "spatial_metadata_editor_visible": true, "spatial_region_label_attribute_id": "",
"gtimeline_visible_row_count": "4"}}, "attribute": {"1": {"aname": "head", "type": 2, "options": {"0": "talk", "1": "watch"}, "default_option_id": "0",
"anchor_id": "FILE1_Z0_XY1", "desc": ""}, "2": {"aname": "body", "type": 2, "options": {"0": "stand", "1": "stoop", "2": "walk"}, "default_option_id": "0",
"anchor_id": "FILE1_Z0_XY1", "desc": ""}, "3": {"aname": "limbs", "type": 2, "options": {"0": "take off hook", "1": "catch"}, "default_option_id": "0",
"anchor_id": "FILE1_Z0_XY1", "desc": ""}}, "file": {"1": {"fname": "1_000061.jpg", "type": 2, "fid": "1", "loc": 1, "src": ""}, "2": {"fname": "1_000091.jpg",
"type": 2, "fid": "2", "loc": 1, "src": ""}, "3": {"fname": "1_000121.jpg", "type": 2, "fid": "3", "loc": 1, "src": ""}, "4": {"fname": "1_000151.jpg",
"type": 2, "fid": "4", "loc": 1, "src": ""}, "5": {"fname": "1_000181.jpg", "type": 2, "fid": "5", "loc": 1, "src": ""}, "6": {"fname": "1_000211.jpg",
"type": 2, "fid": "6", "loc": 1, "src": ""}, "7": {"fname": "1_000241.jpg", "type": 2, "fid": "7", "loc": 1, "src": ""}, "8": {"fname": "1_000271.jpg",
"type": 2, "fid": "8", "loc": 1, "src": ""}, "9": {"fname": "1_000301.jpg", "type": 2, "fid": "9", "loc": 1, "src": ""}, "10": {"fname": "1_000331.jpg",
"type": 2, "fid": "10", "loc": 1, "src": ""}, "11": {"fname": "1_000361.jpg", "type": 2, "fid": "11", "loc": 1, "src": ""}, "12": {"fname": "1_000391.jpg",
"type": 2, "fid": "12", "loc": 1, "src": ""}, "13": {"fname": "1_000421.jpg", "type": 2, "fid": "13", "loc": 1, "src": ""}, "14": {"fname": "1_000451.jpg",
"type": 2, "fid": "14", "loc": 1, "src": ""}, "15": {"fname": "1_000481.jpg", "type": 2, "fid": "15", "loc": 1, "src": ""}, "16": {"fname": "1_000511.jpg",
"type": 2, "fid": "16", "loc": 1, "src": ""}, "17": {"fname": "1_000541.jpg", "type": 2, "fid": "17", "loc": 1, "src": ""}, "18": {"fname": "1_000571.jpg",
"type": 2, "fid": "18", "loc": 1, "src": ""}, "19": {"fname": "1_000601.jpg", "type": 2, "fid": "19", "loc": 1, "src": ""}, "20": {"fname": "1_000631.jpg",
"type": 2, "fid": "20", "loc": 1, "src": ""}, "21": {"fname": "1_000661.jpg", "type": 2, "fid": "21", "loc": 1, "src": ""}, "22": {"fname": "1_000691.jpg",
"type": 2, "fid": "22", "loc": 1, "src": ""}, "23": {"fname": "1_000721.jpg", "type": 2, "fid": "23", "loc": 1, "src": ""}, "24": {"fname": "1_000751.jpg",
"type": 2, "fid": "24", "loc": 1, "src": ""}, "25": {"fname": "1_000781.jpg", "type": 2, "fid": "25", "loc": 1, "src": ""}, "26": {"fname": "1_000811.jpg",
"type": 2, "fid": "26", "loc": 1, "src": ""}, "27": {"fname": "1_000841.jpg", "type": 2, "fid": "27", "loc": 1, "src": ""}, "28": {"fname": "1_000871.jpg",
"type": 2, "fid": "28", "loc": 1, "src": ""}, "29": {"fname": "1_000901.jpg", "type": 2, "fid": "29", "loc": 1, "src": ""}, "30": {"fname": "1_000931.jpg",
"type": 2, "fid": "30", "loc": 1, "src": ""}, "31": {"fname": "1_000961.jpg", "type": 2, "fid": "31", "loc": 1, "src": ""}, "32": {"fname": "1_000991.jpg",
"type": 2, "fid": "32", "loc": 1, "src": ""}, "33": {"fname": "1_001021.jpg", "type": 2, "fid": "33", "loc": 1, "src": ""}, "34": {"fname": "1_001051.jpg",
"type": 2, "fid": "34", "loc": 1, "src": ""}, "35": {"fname": "1_001081.jpg", "type": 2, "fid": "35", "loc": 1, "src": ""}, "36": {"fname": "1_001111.jpg",
"type": 2, "fid": "36", "loc": 1, "src": ""}, "37": {"fname": "1_001141.jpg", "type": 2, "fid": "37", "loc": 1, "src": ""}, "38": {"fname": "1_001171.jpg",
"type": 2, "fid": "38", "loc": 1, "src": ""}, "39": {"fname": "1_001201.jpg", "type": 2, "fid": "39", "loc": 1, "src": ""}, "40": {"fname": "1_001231.jpg",
"type": 2, "fid": "40", "loc": 1, "src": ""}, "41": {"fname": "1_001261.jpg", "type": 2, "fid": "41", "loc": 1, "src": ""}, "42": {"fname": "1_001291.jpg",
"type": 2, "fid": "42", "loc": 1, "src": ""}, "43": {"fname": "1_001321.jpg", "type": 2, "fid": "43", "loc": 1, "src": ""}}, "metadata": {"image1_1": {"vid":
"1", "xy": [2, 689.28384, 28                                                                          g": 0, "z": []}, "image1_2": {"vid": "1",
"xy": [2, 9.40992000000005,                                                                             0, "z": []}, "image1_3": {"vid": "1", "xy":
[2, 278.29247999999995, 349.74233999999996, 231.16032000000007, 339.71724000000001], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []}, "image1_4": {"vid":
"1", "xy": [2, 1659.31008, 656.22906, 248.30207999999993, 354.32964000000004], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []}, "image2_1": {"vid": "2",
"xy": [2, 683.7283199999999, 251.50553999999997, 392.1350399999999, 678.6536399999999], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []}, "image2_2":
{"vid": "2", "xy": [2, 1641.1833600000002, 612.7358399999999, 255.21791999999982, 398.95848], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []},
"image2_3": {"vid": "2", "xy": [2, 8.37312000000007, 617.6001600000001, 347.47776, 434.98511999999994], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z":
                }
#len_x与循环的作用主要是获取每个视频下视频帧的数量
dirname = ''
len_x = {}
for i in info:
    temp_dirname = i.split(',')[0]
    if dirname == temp_dirname:
        #正在循环一个视频文件里的东西
        len_x[dirname] = len_x[dirname] + 1
    else:
        #进入下一个视频文件
        dirname = temp_dirname
        len_x[dirname] = 1
dirname = ''
for i in info:
    temp_dirname = i.split(',')[0]
    if dirname == temp_dirname:
        #正在循环一个视频文件里的东西

        #图片ID从1开始计算
        image_id = image_id + 1
        files_img_num = int(i.split(',')[1])

        # 如果当前出现 files_img_num - 1 与 image_id 不相等的情况
        # 那就代表当前 image_id对应的图片中没有人
        # 那么via的标注记为空
        if files_img_num - 1 != image_id:
            files_dict[str(image_id)] = dict(fname=i.split(',')[0] + '_' + (str((image_id+1)*30+1)).zfill(6) + '.jpg', type=2)
            via3.dumpFiles(files_dict)
            if files_img_num - 1 != image_id:
                while image_id < files_img_num - 1:
                    image_id = image_id + 1
                    files_dict[str(image_id)] = dict(fname=i.split(',')[0] + '_' + (str((image_id+1)*30+1)).zfill(6) + '.jpg', type=2)
                    via3.dumpFiles(files_dict)
                    print("middle loss",image_id,"   ",num_images)
                    print("files_img_num-1",files_img_num-1," image_id",image_id)
                    len_x[dirname] = len_x[dirname] + 1
                    continue
        files_dict[str(image_id)] = dict(fname=i.split(',')[0] + '_' + (str(int(i.split(',')[1])*30+1)).zfill(6) + '.jpg', type=2)

        for vid,result in enumerate(info[i],1):
            xyxy = result
            xyxy[0] = img_W*xyxy[0]
            xyxy[2] = img_W*xyxy[2]
            xyxy[1] = img_H*xyxy[1]
            xyxy[3] = img_H*xyxy[3]
            temp_w = xyxy[2] - xyxy[0]
            temp_h = xyxy[3] - xyxy[1]

            metadata_dict = dict(vid=str(image_id),
                                 xy=[2, float(xyxy[0]), float(xyxy[1]), float(temp_w), float(temp_h)],
                                 av={'1': '0'})

            metadatas_dict['image{}_{}'.format(image_id,vid)] = metadata_dict

        via3.dumpFiles(files_dict)
        via3.dumpMetedatas(metadatas_dict)

        print("OK ",image_id,"   ",num_images)
        if image_id == num_images:
            views_dict = {}
            for i, vid in enumerate(vid_list,1):
                views_dict[vid] = defaultdict(list)
                views_dict[vid]['fid_list'].append(str(i))
            via3.dumpViews(views_dict)
            via3.dempJsonSave()
            print("save")

        #当一个视频的图片的标注信息遍历完后后: image_id == len_x[dirname],
        #但是视频的标注信息长度仍然小于视频实际图片长度
        #即视频图片最后几张都是没有人，导致视频标注信息最后几张没有
        #那么就执行下面的语句，给最后几张图片添加空的标注信息
        print("image_id",image_id," len_x[dirname]",len_x[dirname]," num_images",num_images)
        if image_id == len_x[dirname] and image_id < num_images:
            while image_id < num_images:
                image_id = image_id + 1
                files_dict[str(image_id)] = dict(fname=i.split(',')[0] + '_' + (str((image_id+1)*30+1)).zfill(6) + '.jpg', type=2)
                via3.dumpFiles(files_dict)
            print("end loss",image_id,"   ",num_images)
            views_dict = {}
            for i, vid in enumerate(vid_list,1):
                views_dict[vid] = defaultdict(list)
                views_dict[vid]['fid_list'].append(str(i))
            via3.dumpViews(views_dict)
            via3.dempJsonSave()
            print("save")
    else:
        #进入下一个视频文件
        dirname = temp_dirname
        print("dirname",dirname)

        #为每一个视频文件创建一个via的json文件
        temp_json_path = json_path + dirname + '/' + dirname + '_proposal.json'

        # 获取视频有多少个帧
        for root, dirs, files in os.walk(json_path + dirname, topdown=False):
            if "ipynb_checkpoints" in root:
                continue
            num_images = 0
            for file in files:
                if '.jpg' in file:
                    num_images = num_images + 1
                    temp_img_path = json_path + dirname +'/' + file #图片路径
                    img = cv2.imread(temp_img_path)  #读取图片信息
                    sp = img.shape #[高|宽|像素值由三种原色构成]
                    img_H = sp[0]
                    img_W = sp[1]
        via3 = Via3Json(temp_json_path, mode='dump')
        vid_list = list(map(str,range(1, num_images+1)))
        via3.dumpPrejects(vid_list)
        via3.dumpConfigs()
        via3.dumpAttributes(attributes_dict)


        files_dict,  metadatas_dict = {},{}
        #图片ID从1开始计算
        image_id = 1
```



出于性能原因，未对长行进行解析。解析长度阈值可通过"editor.maxTokenizationLineLength"进行配置。

```
"type": 2, "fid": "42", "loc": 1, "src": ""}, "43": {"fname": "1_001321.jpg", "type": 2, "fid": "43", "loc": 1, "src": ""}}, "metadata": {"image1_1": {"vid":
"1", "xy": [2, 689.28384, 28                                                                g": 0, "z": []}, "image1_2": {"vid": "1",
"xy": [2, 9.409920000000005,                                                                0, "z": []}, "image1_3": {"vid": "1", "xy":
[2, 278.29247999999995, 349.74233999999996, 231.16032000000007, 339.71724000000001], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []}, "image1_4": {"vid":
"1", "xy": [2, 1659.31008, 656.22906, 248.30207999999993, 354.32964000000004], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []}, "image2_1": {"vid": "2",
"xy": [2, 683.7283199999999, 251.50553999999997, 392.1350399999999, 678.6536399999999], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []}, "image2_2":
{"vid": "2", "xy": [2, 1641.1833600000002, 612.7358399999999, 255.21791999999982, 398.95848], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []},
"image2_3": {"vid": "2", "xy": [2, 8.373120000000007, 617.6001600000001, 347.47776, 434.98511999999994], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z":
[]}, "image2_4": {"vid": "2", "xy": [2, 278.33184, 342.39023999999995, 230.83584000000002, 347.81400000000001], "av": {"1": "", "2": "", "3": ""}, "flg": 0,
"z": []}, "image3_1": {"vid": "3", "xy": [2, 626.23488, 320.16492, 420.58752000000004, 608.4612], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []},
"image3_2": {"vid": "3", "xy": [2, 1631.3462399999999, 609.64596, 269.63519999999994, 407.11032], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []},
"image3_3": {"vid": "3", "xy": [2, 1.978560000000007, 620.14734, 248.9376, 442.19843999999999], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []},
"image4_1": {"vid": "4", "xy": [2, 2.822399999999982, 618.20334, 287.24352, 458.89956000000006], "av": {"1": "", "2": "", "3": ""}, "flg": 0, "z": []},
"image4_2": {"vid": "4", "xy": [2, 285.98879999999997, 327.87989999999996, 215.79839999999996, 349.94268000000005], "av": {"1": "", "2": "", "3": ""}, "flg":
0, "z": []}, "image4_3": {"vid": "4", "xy": [2, 617.12543999999999, 312.31925999999993, 429.05280000000005, 612.88380000000001], "av": {"1": "", "2": "", "3":
    files_dict[str(image_id)] = dict(fname=i.split(',')[0] + '_' + (str(int(i.split(',')[1])*30+1)).zfill(6) + '.jpg', type=2)

    for vid,result in enumerate(info[i],1):
        xyxy = result
        xyxy[0] = img_W*xyxy[0]
        xyxy[2] = img_W*xyxy[2]
        xyxy[1] = img_H*xyxy[1]
        xyxy[3] = img_H*xyxy[3]
        temp_w = xyxy[2] - xyxy[0]
        temp_h = xyxy[3] - xyxy[1]

        metadata_dict = dict(vid=str(image_id),
                             xy=[2, float(xyxy[0]), float(xyxy[1]), float(temp_w), float(temp_h)],
                             av={'1': '0'})
        #print(metadata_dict)
        metadatas_dict['image{}_{}'.format(image_id,vid)] = metadata_dict

    via3.dumpFiles(files_dict)
    via3.dumpMetedatas(metadatas_dict)
```

3.4 去掉via默认值

标注时有默认值，这个会影响我们的标注，需要取消掉。

生成的标注文件保存在：Dataset/choose_frames_middle中1.proposal_s.json

3.5使用via进行标注

下载via标注工具后

然后使用via进行标注

via官网：

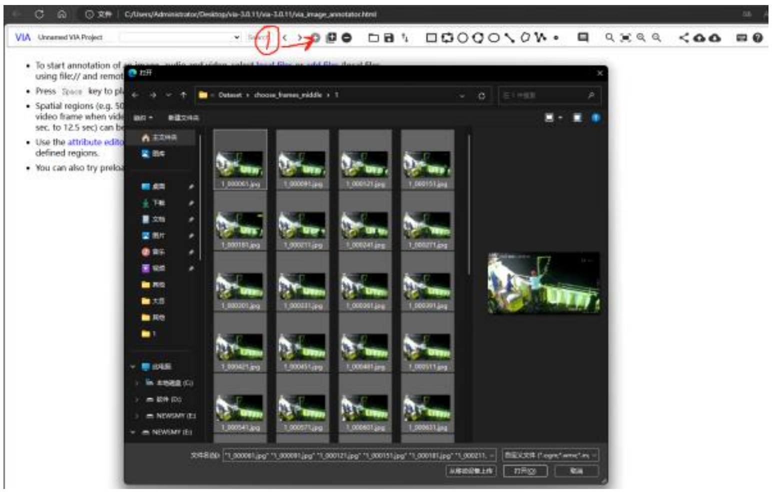
https://www.robots.ox.ac.uk/~vgg/software/via/

via标注工具下载链接：

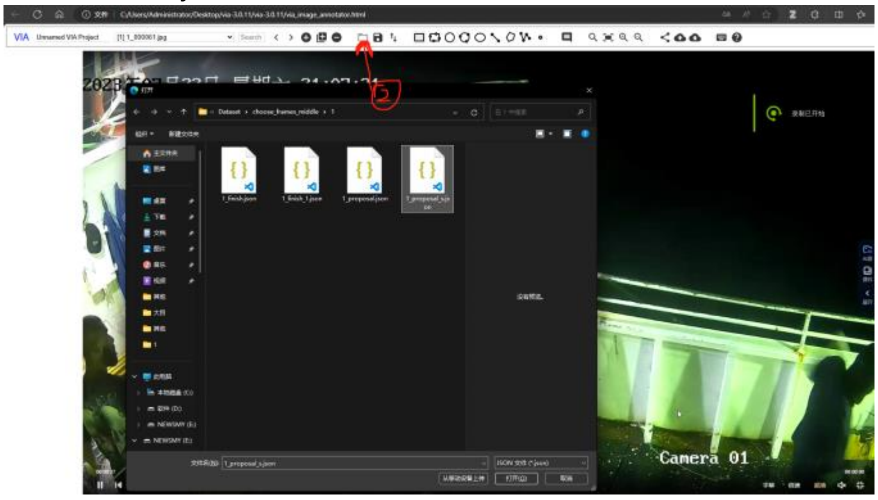
https://www.robots.ox.ac.uk/~vgg/software/via/downloads/via3/via-3.0.11.zip

点击 via_image_annotator.html

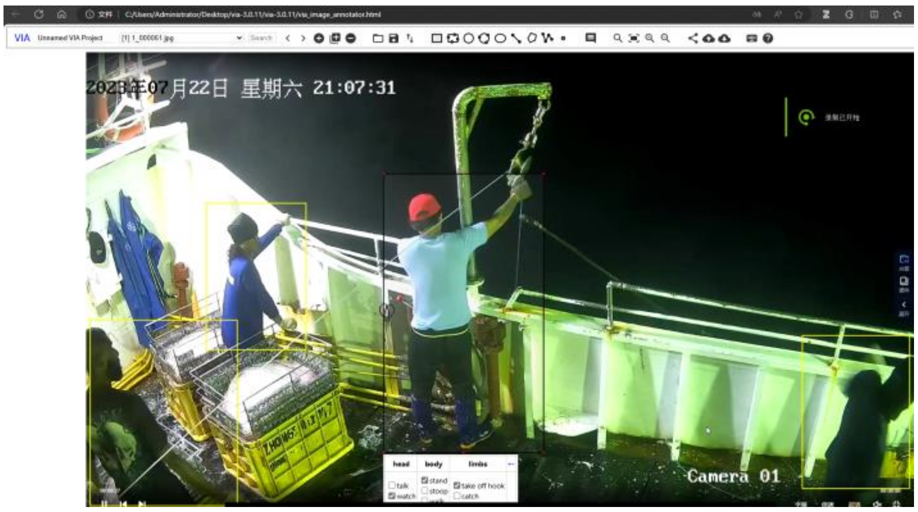
1、选择对应的需要标注的图片（不需要全部标注）



2、选择上一步生成的.json文件



3、进行标注，最后保存为1_finish.json文件


C:/Users/Administrator/Desktop/via-3.0.11/via-3.0.11/via_image_annotator.html
VIA
Unnamed VIA Project
[1] 1_000061.jpg
Search
2023年07月22日 星期六 21:07:31
Camera 01

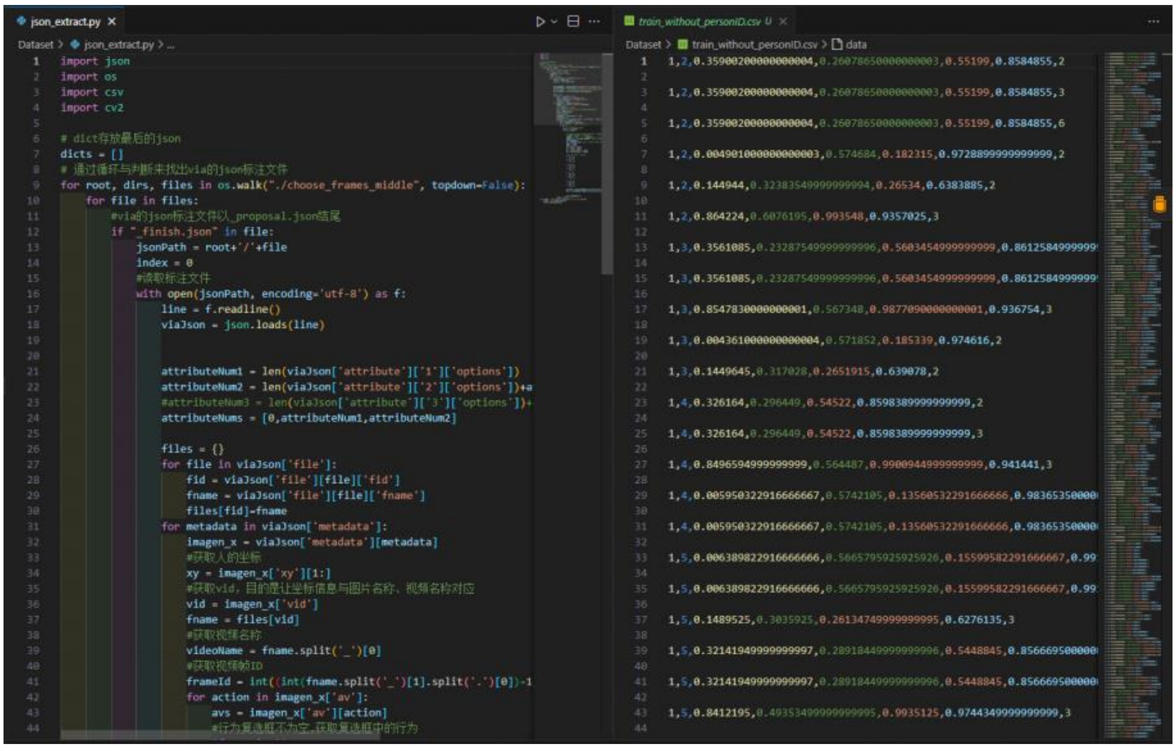
head	body	limbs
☐ talk	☑ stand	☑ take off hook
☑ watch	☐ stoop	☐ catch
	☐ walk	

4 via信息提取

2024年8月10日 18:05

4.1运行json_extract.py程序，提取出上一步的标注信息文件，保存为.csv文件（还缺少人的ID编号）

会在Dataset/下生成：train_without_personID.csv


json_extract.py
Dataset > json_extract.py > ...

```python
import json
import os
import csv
import cv2

# dict存放最后的json
dicts = []
# 通过循环与判断来找出via的json标注文件
for root, dirs, files in os.walk("./choose_frames_middle", topdown=False):
    for file in files:
        #via的json标注文件以_proposal.json结尾
        if "_finish.json" in file:
            jsonPath = root+'/'+file
            index = 0
            #读取标注文件
            with open(jsonPath, encoding='utf-8') as f:
                line = f.readline()
                viaJson = json.loads(line)


                attributeNum1 = len(viaJson['attribute']['1']['options'])
                attributeNum2 = len(viaJson['attribute']['2']['options'])+a
                #attributeNum3 = len(viaJson['attribute']['3']['options'])+
                attributeNums = [0,attributeNum1,attributeNum2]

                files = {}
                for file in viaJson['file']:
                    fid = viaJson['file'][file]['fid']
                    fname = viaJson['file'][file]['fname']
                    files[fid]=fname
                for metadata in viaJson['metadata']:
                    imagen_x = viaJson['metadata'][metadata]
                    #获取人的坐标
                    xy = imagen_x['xy'][1:]
                    #获取vid，目的是让坐标信息与图片名称、视频名称对应
                    vid = imagen_x['vid']
                    fname = files[vid]
                    #获取视频名称
                    videoName = fname.split('_')[0]
                    #获取视频帧ID
                    frameId = int((int(fname.split('_')[1].split('.')[0])-1
                    for action in imagen_x['av']:
                        avs = imagen_x['av'][action]
                        #行为复选框不为空，获取复选框中的行为
```

train_without_personID.csv U
Dataset > train_without_personID.csv > data

```
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,2

1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,3

1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,6

1,2,0.004901000000000003,0.574684,0.182315,0.9728899999999999,2

1,2,0.144944,0.32383549999999994,0.26534,0.6383885,2

1,2,0.864224,0.6076195,0.993548,0.9357025,3

1,3,0.3561085,0.23287549999999996,0.5603454999999999,0.86125849999999

1,3,0.3561085,0.23287549999999996,0.5603454999999999,0.86125849999999

1,3,0.8547830000000001,0.567348,0.9877090000000001,0.936754,3

1,3,0.004361000000000004,0.571852,0.185339,0.974616,2

1,3,0.1449645,0.317028,0.2651915,0.639078,2

1,4,0.326164,0.296449,0.54522,0.8598389999999999,2

1,4,0.326164,0.296449,0.54522,0.8598389999999999,3

1,4,0.8496594999999999,0.564487,0.9900944999999999,0.941441,3

1,4,0.005950322916666667,0.5742105,0.13560532291666666,0.98365350000

1,4,0.005950322916666667,0.5742105,0.13560532291666666,0.98365350000

1,5,0.006389822916666666,0.5665795925925926,0.15599582291666667,0.99

1,5,0.006389822916666666,0.5665795925925926,0.15599582291666667,0.99

1,5,0.1489525,0.3035925,0.26134749999999995,0.6276135,3

1,5,0.32141949999999997,0.28918449999999996,0.5448845,0.856669500000

1,5,0.32141949999999997,0.28918449999999996,0.5448845,0.856669500000

1,5,0.8412195,0.49353499999999995,0.9935125,0.9744349999999999,3

```

命令行

cd ava_made/Dataset

python json_extract.py

生成的.csv文件：
第一列为视频ID
第二列为视频的秒数
第三-六列为人的坐标信息
第七列为标注的动作类型

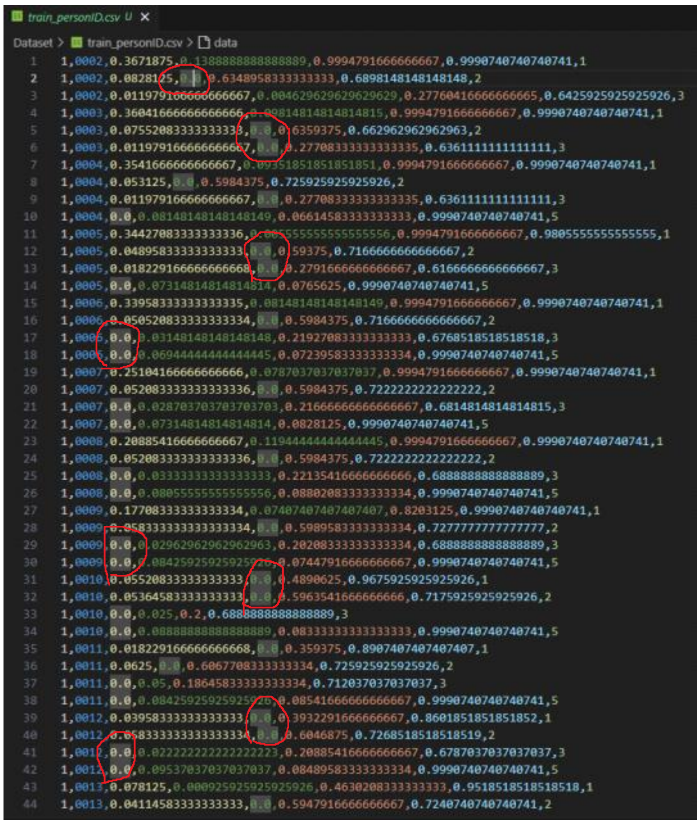
5 deep sort检测ID（error）

2024年8月10日 18:13

使用YOLO与deep结合，对YOLO检测出来的人进行标号，生成带有坐标信息与标号的.csv文件

生成的.csv文件:
第一列为视频ID
第二列为视频的秒数
第三-六列为人的坐标信息
第七列为人的标号

```python
import argparse
import os
import csv
import torch
import numpy as np
import pickle
from PIL import Image
from v10.ultralytics import YOLO  # 导入YOLOv10
from deep_sort_pytorch.utils.parser import get_config
from deep_sort_pytorch.deep_sort import DeepSort
# dict存放最后的json
dicts = []
def detect(opt):
    source = opt.source

    # 加载Deep SORT的配置
    cfg = get_config()
    cfg.merge_from_file(opt.config_deepsort)
    # 加载自训练的YOLOv10模型
    model = YOLO('v10/runs/train/frames/weights/best.pt')  # 你可以根据需要更换模型权重文件
    # 加载目标检测提案数据
    f = open('./mywork/dense_proposals_train_deepsort.pkl', 'rb')
    info = pickle.load(f, encoding='iso-8859-1')

    tempFileName = ''

    # 设置YOLOv10的置信度阈值
    confidence_threshold = 0.6

    for i in info:
        dets = info[i]
        tempName = i.split(',')

        if tempName[0] != tempFileName:
            deepsort = DeepSort(cfg.DEEPSORT.REID_CKPT,
                                max_dist=cfg.DEEPSORT.MAX_DIST, min_confidence=cfg.DEEPSORT.MIN_CONFIDENCE,
                                max_iou_distance=cfg.DEEPSORT.MAX_IOU_DISTANCE,
                                max_age=cfg.DEEPSORT.MAX_AGE, n_init=cfg.DEEPSORT.N_INIT, nn_budget=cfg.DEEPSORT.NN_BUDGET,
                                use_cuda=True)
            tempFileName = tempName[0]

        # 使用os.path.join拼接路径
        imOPath = os.path.join(source, tempName[0], f"{tempName[0]}_{str(int(tempName[1])*30+1).zfill(6)}.jpg")
        print(f"Attempting to load image from path: {imOPath}")
        # 确认文件存在
        if not os.path.exists(imOPath):
            raise FileNotFoundError(f"Image not found at path: {imOPath}")
        im0 = np.array(Image.open(imOPath))
        # YOLOv10 推理，使用置信度阈值过滤检测框
        results = model.predict(imOPath, conf=confidence_threshold)

        # 检查YOLOv10检测结果是否为空
        if not results or len(results[0].boxes) == 0:
            print(f"No valid detections in image: {imOPath}")
            continue

        # 提取YOLOv10的检测结果
        dets = results[0].boxes
        xyxys = dets.xyxy.cpu().numpy()
        confs = dets.conf.cpu().numpy()
        clss = dets.cls.cpu().numpy()

        # 输出检测结果进行调试
        print(f"Detections for {imOPath}: xyxys={xyxys}, confs={confs}, clss={clss}")

        # Deep SORT 跟踪更新
        outputs = deepsort.update(xyxys, confs, clss, im0)

        if len(outputs) > 0:
            for output in outputs:
                x1 = output[0] / im0.shape[1]
                y1 = output[1] / im0.shape[0]
                x2 = output[2] / im0.shape[1]
                y2 = output[3] / im0.shape[0]
                dict_entry = [tempName[0], tempName[1], x1, y1, x2, y2, output[4]]
                dicts.append(dict_entry)

        # 保存结果到CSV文件
        with open('../Dataset/train_personID.csv', "w", newline='') as csvfile:
            writer = csv.writer(csvfile)
            writer.writerows(dicts)
if __name__ == '__main__':
    parser = argparse.ArgumentParser()

    parser.add_argument('--deep_sort_weights', type=str, default='deep_sort_pytorch/deep_sort/deep/checkpoint/ckpt.t7', help='ckpt.t7 path')
    parser.add_argument('--source', type=str, default='0', help='source')
    parser.add_argument('--save-txt', action='store_true', help='save MOT compliant results to *.txt')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or --class 16 17')
    parser.add_argument("--config_deepsort", type=str, default="deep_sort_pytorch/configs/deep_sort.yaml")

    opt = parser.parse_args()
    with torch.no_grad():
        detect(opt)
```



```
1,0002,0.3671875,0.1388888888888889,0.9994791666666667,0.9990740740740741,1
1,0002,0.0828125,0.0,0.6348958333333333,0.6898148148148148,2
1,0002,0.011979166666666667,0.004629629629629629,0.27760416666666665,0.6425925925925926,3
1,0003,0.36041666666666666,0.00814814814814815,0.9994791666666667,0.9990740740740741,1
1,0003,0.07552083333333333,0.0,0.6359375,0.662962962962963,2
1,0003,0.011979166666666667,0.0,0.27708333333333335,0.6361111111111111,3
1,0004,0.3541666666666667,0.09351851851851851,0.9994791666666667,0.9990740740740741,1
1,0004,0.053125,0.0,0.5984375,0.725925925925926,2
1,0004,0.011979166666666667,0.0,0.27708333333333335,0.6361111111111111,3
1,0004,0.0,0.0814814814814814 9,0.06614583333333333,0.9990740740740741,5
1,0005,0.34427083333333336,0.005555555555555556,0.9994791666666667,0.9805555555555555,1
1,0005,0.04895833333333333,0.0,0.59375,0.7166666666666667,2
1,0005,0.018229166666666668,0.0,0.2791666666666667,0.6166666666666667,3
1,0005,0.0,0.0731481481481481 4,0.0765625,0.9990740740740741,5
1,0006,0.33958333333333335,0.0814814814814814 9,0.9994791666666667,0.9990740740740741,1
1,0006,0.05052083333333334,0.0,0.5984375,0.7166666666666667,2
1,0006,0.0,0.0314814814814814 8,0.21927083333333333,0.6768518518518518,3
1,0006,0.0,0.06944444444444445,0.07239583333333334,0.9990740740740741,5
1,0007,0.25104166666666666,0.0787037037037037,0.9994791666666667,0.9990740740740741,1
1,0007,0.05208333333333336,0.0,0.5984375,0.7222222222222222,2
1,0007,0.0,0.028703703703703703,0.21666666666666667,0.6814814814814815,3
1,0007,0.0,0.0731481481481481 4,0.0828125,0.9990740740740741,5
1,0008,0.20885416666666667,0.11944444444444445,0.9994791666666667,0.9990740740740741,1
1,0008,0.05208333333333336,0.0,0.5984375,0.7222222222222222,2
1,0008,0.0,0.03333333333333333,0.22135416666666666,0.6888888888888889,3
1,0008,0.0,0.08055555555555556,0.08802083333333334,0.9990740740740741,5
1,0009,0.17708333333333334,0.07407407407407407,0.8203125,0.9990740740740741,1
1,0009,0.05833333333333334,0.0,0.5989583333333334,0.7277777777777777,2
1,0009,0.0,0.02962962962962963,0.20208333333333334,0.6888888888888889,3
1,0009,0.0,0.08425925925925926,0.07447916666666667,0.9990740740740741,5
1,0010,0.05520833333333333,0.0,0.4890625,0.9675925925925926,1
1,0010,0.05364583333333333,0.0,0.5963541666666666,0.7175925925925926,2
1,0010,0.0,0.025,0.2,0.6888888888888889,3
1,0010,0.0,0.08888888888888889,0.08333333333333333,0.9990740740740741,5
1,0011,0.018229166666666668,0.0,0.359375,0.8907407407407407,1
1,0011,0.0625,0.0,0.6067708333333334,0.725925925925926,2
1,0011,0.0,0.05,0.18645833333333334,0.7120370370370370 37,3
1,0011,0.0,0.08425925925925926,0.08541666666666667,0.9990740740740741,5
1,0012,0.03958333333333333,0.0,0.3932291666666667,0.8601851851851852,1
1,0012,0.05833333333333334,0.0,0.6046875,0.7268518518518519,2
1,0012,0.022222222222222223,0.20885416666666667,0.6787037037037037,3
1,0012,0.0,0.09537037037037037,0.08489583333333334,0.9990740740740741,5
1,0013,0.078125,0.000925925925925926,0.4630208333333333,0.9518518518518518,1
1,0013,0.04114583333333333,0.0,0.5947916666666667,0.7240740740740741,2
```

5 deep sort检测ID

2024年8月27日　　9:34

5.1 dense_proposals_train_deepsort.py

由于deepsort需要提前送入2帧图片，然后才能从第三帧开始标注人的ID

dense_proposals_train.pkl(第三步生成的)是从第三张开始的（即缺失了0，1）

所以需要将0，1添加

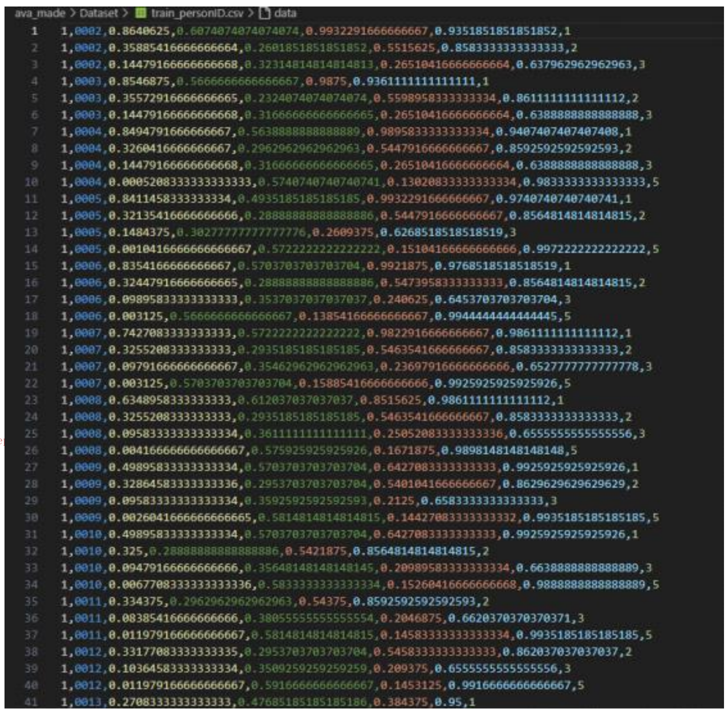
接下来使用deep sort来关联人的ID

将图片与yolov10检测出来的坐标，送入deep sort进行检测

命令行

```
cd  ava_made/Dataset/yolovDeepsort/mywork

python
dense_proposals_train_deepsort.py ../v10/runs/detect/exp/labels ./dense_proposals_train_
deepsort.pkl show

cd  ava_made/Dataset/yolovDeepsort

python yolov10_to_deepsort.py --source D:/github_files/slowfast-mmaction-
ava/ava_made/Dataset/frames
```

改进优化后的代码，生成的csv文件如右图，可以看出没有再次出现坐标为0.0的错误现象



```
ava_made > Dataset > train_personID.csv > data
1,0002,0.8640625,0.6074074074074074,0.9932291666666667,0.9351851851851852,1
1,0002,0.35885416666666664,0.2601851851851852,0.5515625,0.8583333333333333,2
1,0002,0.14479166666666668,0.32314814814814813,0.26510416666666664,0.637962962962963,3
1,0003,0.8546875,0.5666666666666667,0.9875,0.9361111111111111,1
1,0003,0.35572916666666665,0.2324074074074074,0.5598958333333334,0.8611111111111112,2
1,0003,0.14479166666666668,0.31666666666666665,0.26510416666666664,0.6388888888888888,3
1,0004,0.8494791666666667,0.5638888888888889,0.9895833333333334,0.9407407407407408,1
1,0004,0.3260416666666667,0.2962962962962963,0.5447916666666667,0.8592592592592593,2
1,0004,0.14479166666666668,0.31666666666666665,0.26510416666666664,0.6388888888888888,3
1,0004,0.0005208333333333333,0.5740740740740741,0.13020833333333334,0.9833333333333333,5
1,0005,0.8411458333333334,0.4935185185185185,0.9932291666666667,0.9740740740740741,1
1,0005,0.32135416666666666,0.28888888888888886,0.5447916666666667,0.8564814814814815,2
1,0005,0.1484375,0.30277777777777776,0.2609375,0.6268518518518519,3
1,0005,0.0010416666666666667,0.5722222222222222,0.15104166666666666,0.9972222222222222,5
1,0006,0.8354166666666667,0.5703703703703704,0.9921875,0.9768518518518519,1
1,0006,0.32447916666666665,0.28888888888888886,0.5473958333333333,0.8564814814814815,2
1,0006,0.09895833333333333,0.3537037037037037,0.240625,0.6453703703703704,3
1,0006,0.003125,0.5666666666666667,0.13854166666666667,0.9944444444444445,5
1,0007,0.7427083333333333,0.5722222222222222,0.9822916666666667,0.9861111111111112,1
1,0007,0.3255208333333333,0.2935185185185185,0.5463541666666667,0.8583333333333333,2
1,0007,0.09791666666666667,0.3546296296296296,0.23697916666666666,0.6527777777777778,3
1,0007,0.003125,0.5703703703703704,0.15885416666666666,0.9925925925925926,5
1,0008,0.6348958333333333,0.612037037037037,0.8515625,0.9861111111111112,1
1,0008,0.3255208333333333,0.2935185185185185,0.5463541666666667,0.8583333333333333,2
1,0008,0.09583333333333334,0.3611111111111111,0.25052083333333336,0.6555555555555556,3
1,0008,0.004166666666666667,0.575925925925926,0.1671875,0.9898148148148148,5
1,0009,0.49895833333333334,0.5703703703703704,0.6427083333333333,0.9925925925925926,1
1,0009,0.32864583333333336,0.2953703703703704,0.5401041666666667,0.8629629629629629,2
1,0009,0.09583333333333334,0.3592592592592593,0.2125,0.6583333333333333,3
1,0009,0.0026041666666666665,0.5814814814814815,0.14427083333333332,0.9935185185185185,5
1,0010,0.49895833333333334,0.5703703703703704,0.6427083333333333,0.9925925925925926,1
1,0010,0.325,0.28888888888888886,0.5421875,0.8564814814814815,2
1,0010,0.09479166666666666,0.35648148148148145,0.20989583333333334,0.6638888888888889,3
1,0010,0.006770833333333336,0.5833333333333334,0.15260416666666668,0.9888888888888889,5
1,0011,0.334375,0.2962962962962963,0.54375,0.8592592592592593,2
1,0011,0.08385416666666666,0.3805555555555554,0.2046875,0.6620370370370371,3
1,0011,0.011979166666666667,0.5814814814814815,0.14583333333333334,0.9935185185185185,5
1,0012,0.33177083333333335,0.2953703703703704,0.5458333333333333,0.862037037037037,2
1,0012,0.10364583333333334,0.3509259259259259,0.209375,0.6555555555555556,3
1,0012,0.011979166666666667,0.5916666666666667,0.1453125,0.9916666666666667,5
1,0013,0.2708333333333333,0.47685185185185186,0.384375,0.95,1
```

```python
import argparse
import csv
import os
import torch
import numpy as np
import pickle
from PIL import Image
from v10.ultralytics import YOLO  # 替换为 YOLOv10
from deep_sort_pytorch.utils.parser import get_config
from deep_sort_pytorch.deep_sort import DeepSort
# 手动实现 xyxy2xywh 函数
def xyxy2xywh(x):
    # 将(x1, y1, x2, y2)转换为(x_center, y_center, width, height)
    y = torch.zeros_like(x)
    y[:, 0] = (x[:, 0] + x[:, 2]) / 2  # x_center
    y[:, 1] = (x[:, 1] + x[:, 3]) / 2  # y_center
    y[:, 2] = x[:, 2] - x[:, 0]        # width
    y[:, 3] = x[:, 3] - x[:, 1]        # height
    return y
# dict存放最后的json
dicts = []
def detect(opt):
    source = opt.source

    # 加载 Deep SORT 配置
    cfg = get_config()
    cfg.merge_from_file(opt.config_deepsort)

    # 加载 YOLOv10 模型
    model = YOLO(r'C:/Users/Administrator/Desktop/Custom-ava-dataset_Custom-Spatio-Temporally-Action-Video-Dataset/yolovDee

    # 加载目标检测提案数据
    with open('./mywork/dense_proposals_train_deepsort.pkl', 'rb') as f:
        info = pickle.load(f, encoding='iso-8859-1')

    tempFileName = ''

    for i in info:
        dets = info[i]
        tempName = i.split(',')

        # 如果读取到新的文件，重新初始化 DeepSORT
        if tempName[0] != tempFileName:
            deepsort = DeepSort(cfg.DEEPSORT.REID_CKPT,
                                max_dist=cfg.DEEPSORT.MAX_DIST, min_confidence=cfg.DEEPSORT.MIN_CONFIDENCE,
                                max_iou_distance=cfg.DEEPSORT.MAX_IOU_DISTANCE,
                                max_age=cfg.DEEPSORT.MAX_AGE, n_init=cfg.DEEPSORT.N_INIT, nn_budget=cfg.DEEPSORT.NN_BUDGET,
                                use_cuda=True)
            tempFileName = tempName[0]

        # 读取图像并获取尺寸
        im0Path = os.path.join(source, tempName[0], f"{tempName[0]}_{str(int(tempName[1])*30+1).zfill(6)}.jpg")
        im0 = np.array(Image.open(im0Path))
        imgsz = im0.shape

        # YOLOv10 推理
        results = model.predict(im0Path, conf=0.6)

        # 检查检测结果是否为空
        if not results or len(results[0].boxes) == 0:
            print(f"No valid detections in image: {im0Path}")
            continue

        # 提取 YOLOv10 的检测结果
        dets = results[0].boxes
        xyxys = dets.xyxy.cpu().numpy()  # 转换为 numpy 数组
        confs = dets.conf.cpu().numpy()
        clss = np.zeros_like(confs)  # 将类别ID设置为0 (假设全为person)

        # 将 YOLO 的 (x1, y1, x2, y2) 坐标转换为 DeepSORT 所需的 (x_center, y_center, width, height)
        xywhs = xyxy2xywh(torch.FloatTensor(xyxys))

        # Deep SORT 跟踪
        outputs = deepsort.update(xywhs.cpu(), torch.FloatTensor(confs), torch.FloatTensor(clss), im0)

        # 处理 DeepSORT 的跟踪结果
        if len(outputs) > 0:
            for output in outputs:
                x1 = output[0] / imgsz[1]
                y1 = output[1] / imgsz[0]
                x2 = output[2] / imgsz[1]
                y2 = output[3] / imgsz[0]
                dict_entry = [tempName[0], tempName[1], x1, y1, x2, y2, output[4]]
                dicts.append(dict_entry)

        # 保存结果到 CSV 文件
        with open('../Dataset/train_personID.csv', "w", newline='') as csvfile:
            writer = csv.writer(csvfile)
            writer.writerows(dicts)
if __name__ == '__main__':
    parser = argparse.ArgumentParser()

    parser.add_argument('--deep_sort_weights', type=str, default='deep_sort_pytorch/deep_sort/deep/checkpoint/ckpt.t7', help='ckpt.t7 path')
    parser.add_argument('--source', type=str, default='0', help='source')
    parser.add_argument('--save-txt', action='store_true', help='save MOT compliant results to *.txt')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or --class 16 17')
    parser.add_argument("--config_deepsort", type=str, default="deep_sort_pytorch/configs/deep_sort.yaml")
    parser.add_argument('--weights', type=str, help='model weights path')  # 新增的模型权重参数

    opt = parser.parse_args()
    with torch.no_grad():
        detect(opt)
```

相较于之前的代码主要在此处部分做了优化

将 YOLO 的 (x1, y1, x2, y2) 坐标转换为 DeepSORT 所需的 (x_center, y_center, width, height)

Deep SORT 跟踪

处理 DeepSORT 的跟踪结果

6 融合actions与personID（error）

2024年8月10日 18:19

命令行

cd ava_made/Dataset

python train_temp.py

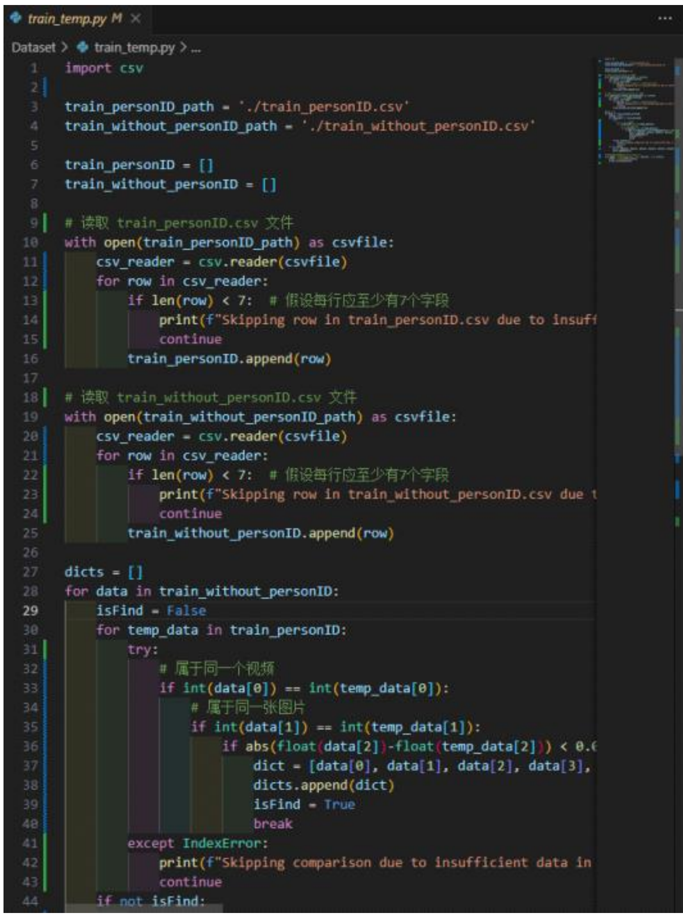
目前已经有2个文件了：

1，train_personID.csv **包含 坐标、personID**

2，train_without_personID.csv **包含 坐标、actions**

所以现在需要将两者拼在一起

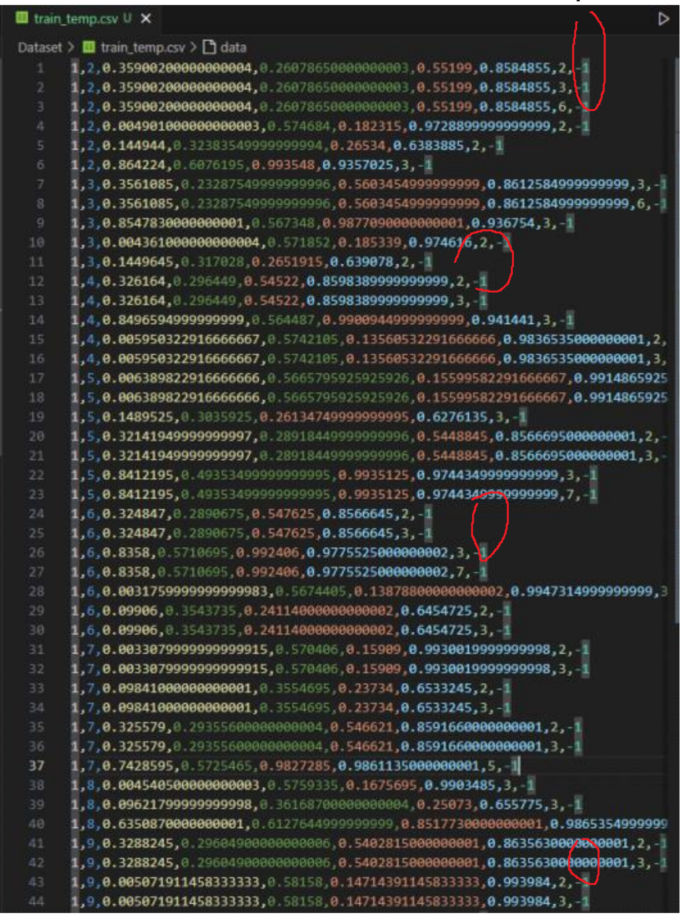
运行结束后，会发现有些ID是-1，这些-1是deepsort未检测出来的数据，原因是人首次出现或者出现时间过短，deepsort未检测出ID。



train_temp.py M

Dataset > train_temp.py > ...

```python
import csv

train_personID_path = './train_personID.csv'
train_without_personID_path = './train_without_personID.csv'

train_personID = []
train_without_personID = []

# 读取 train_personID.csv 文件
with open(train_personID_path) as csvfile:
    csv_reader = csv.reader(csvfile)
    for row in csv_reader:
        if len(row) < 7:  # 假设每行应至少有7个字段
            print(f"Skipping row in train_personID.csv due to insuff
            continue
        train_personID.append(row)

# 读取 train_without_personID.csv 文件
with open(train_without_personID_path) as csvfile:
    csv_reader = csv.reader(csvfile)
    for row in csv_reader:
        if len(row) < 7:  # 假设每行应至少有7个字段
            print(f"Skipping row in train_without_personID.csv due t
            continue
        train_without_personID.append(row)

dicts = []
for data in train_without_personID:
    isFind = False
    for temp_data in train_personID:
        try:
            # 属于同一个视频
            if int(data[0]) == int(temp_data[0]):
                # 属于同一张图片
                if int(data[1]) == int(temp_data[1]):
                    if abs(float(data[2])-float(temp_data[2])) < 0.(
                        dict = [data[0], data[1], data[2], data[3],
                        dicts.append(dict)
                        isFind = True
                        break
        except IndexError:
            print(f"Skipping comparison due to insufficient data in
            continue
    if not isFind:
```



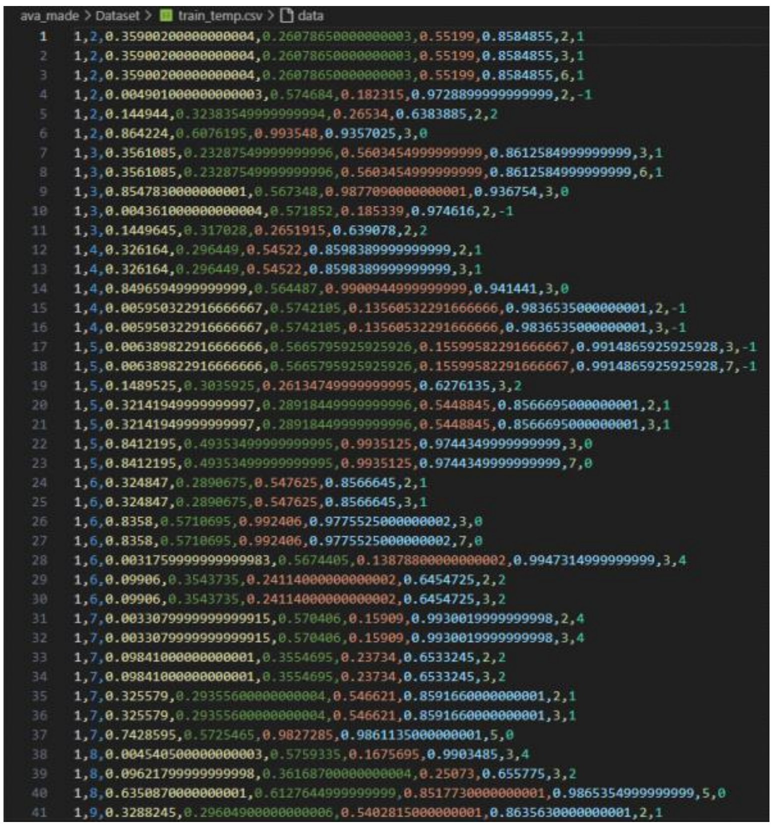
train_temp.csv U

Dataset > train_temp.csv > data

```
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,2,-1
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,3,-1
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,6,-1
1,2,0.004901000000000003,0.574684,0.182315,0.9728899999999999,2,-1
1,2,0.144944,0.32383549999999994,0.26534,0.6383885,2,-1
1,2,0.864224,0.6076195,0.993548,0.9357025,3,-1
1,3,0.3561085,0.23287549999999996,0.5603454999999999,0.8612584999999999,3,-1
1,3,0.3561085,0.23287549999999996,0.5603454999999999,0.8612584999999999,6,-1
1,3,0.8547830000000001,0.567348,0.9877090000000001,0.936754,3,-1
1,3,0.004361000000000004,0.571852,0.185339,0.974616,2,-1
1,3,0.1449645,0.317028,0.2651915,0.639078,2,-1
1,4,0.326164,0.296449,0.54522,0.8598389999999999,2,-1
1,4,0.326164,0.296449,0.54522,0.8598389999999999,3,-1
1,4,0.8496594999999999,0.564487,0.9900944999999999,0.941441,3,-1
1,4,0.005950322916666667,0.5742105,0.13560532291666666,0.9836535000000001,2,
1,4,0.005950322916666667,0.5742105,0.13560532291666666,0.9836535000000001,3,
1,5,0.006389822916666666,0.5665795925925926,0.15599582291666667,0.9914865925
1,5,0.006389822916666666,0.5665795925925926,0.15599582291666667,0.9914865925
1,5,0.1489525,0.3035925,0.26134749999999995,0.6276135,3,-1
1,5,0.32141949999999997,0.28918449999999996,0.5448845,0.8566695000000001,2,-
1,5,0.32141949999999997,0.28918449999999996,0.5448845,0.8566695000000001,3,-
1,5,0.8412195,0.49353499999999995,0.9935125,0.9744349999999999,3,-1
1,5,0.8412195,0.49353499999999995,0.9935125,0.9744349999999999,7,-1
1,6,0.324847,0.2890675,0.547625,0.8566645,2,-1
1,6,0.324847,0.2890675,0.547625,0.8566645,3,-1
1,6,0.8358,0.5710695,0.992406,0.9775525000000002,3,-1
1,6,0.8358,0.5710695,0.992406,0.9775525000000002,7,-1
1,6,0.0031759999999999983,0.5674405,0.1387880000000002,0.9947314999999999,3
1,6,0.09906,0.3543735,0.24114000000000002,0.6454725,2,-1
1,6,0.09906,0.3543735,0.24114000000000002,0.6454725,3,-1
1,7,0.0033079999999999915,0.570406,0.15909,0.9930019999999998,2,-1
1,7,0.0033079999999999915,0.570406,0.15909,0.9930019999999998,3,-1
1,7,0.09841000000000001,0.3554695,0.23734,0.6533245,2,-1
1,7,0.09841000000000001,0.3554695,0.23734,0.6533245,3,-1
1,7,0.325579,0.29355600000000004,0.546621,0.8591660000000001,2,-1
1,7,0.325579,0.29355600000000004,0.546621,0.8591660000000001,3,-1
1,7,0.7428595,0.5725465,0.9827285,0.9861135000000001,5,-1
1,8,0.004540500000000003,0.5759335,0.1675695,0.9903485,3,-1
1,8,0.09621799999999998,0.36168700000000004,0.25073,0.655775,3,-1
1,8,0.6350870000000001,0.6127644999999999,0.8517730000000001,0.9865354999999
1,9,0.3288245,0.29604900000000006,0.5402815000000001,0.8635630000000001,2,-1
1,9,0.3288245,0.29604900000000006,0.5402815000000001,0.8635630000000001,3,-1
1,9,0.005071911458333333,0.58158,0.14714391145833333,0.993984,2,-1
1,9,0.005071911458333333,0.58158,0.14714391145833333,0.993984,3,-1
```

在第五步正确处理后，生成的csv文件，再次融合时便正常了

没有出现全部为“-1”的情况



ava_made > Dataset > train_temp.csv > data

```
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,2,1
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,3,1
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,6,1
1,2,0.004901000000000003,0.574684,0.182315,0.9728899999999999,2,-1
1,2,0.144944,0.32383549999999994,0.26534,0.6383885,2,2
1,2,0.864224,0.6076195,0.993548,0.9357025,3,0
1,3,0.3561085,0.23287549999999996,0.5603454999999999,0.8612584999999999,3,1
1,3,0.3561085,0.23287549999999996,0.5603454999999999,0.8612584999999999,6,1
1,3,0.8547830000000001,0.567348,0.9877090000000001,0.936754,3,0
1,3,0.004361000000000004,0.571852,0.185339,0.974616,2,-1
1,3,0.1449645,0.317028,0.2651915,0.639078,2,2
1,4,0.326164,0.296449,0.54522,0.8598389999999999,2,1
1,4,0.326164,0.296449,0.54522,0.8598389999999999,3,1
1,4,0.8496594999999999,0.564487,0.9900944999999999,0.941441,3,0
1,4,0.005950322916666667,0.5742105,0.13560532291666666,0.9836535000000001,2,-1
1,4,0.005950322916666667,0.5742105,0.13560532291666666,0.9836535000000001,3,-1
1,5,0.006389822916666666,0.5665795925925926,0.15599582291666667,0.9914865925925928,3,-1
1,5,0.006389822916666666,0.5665795925925926,0.15599582291666667,0.9914865925925928,7,-1
1,5,0.1489525,0.3035925,0.26134749999999995,0.6276135,3,2
1,5,0.32141949999999997,0.28918449999999996,0.5448845,0.8566695000000001,2,1
1,5,0.32141949999999997,0.28918449999999996,0.5448845,0.8566695000000001,3,1
1,5,0.8412195,0.49353499999999995,0.9935125,0.9744349999999999,3,0
1,5,0.8412195,0.49353499999999995,0.9935125,0.9744349999999999,7,0
1,6,0.324847,0.2890675,0.547625,0.8566645,2,1
1,6,0.324847,0.2890675,0.547625,0.8566645,3,1
1,6,0.8358,0.5710695,0.992406,0.9775525000000002,3,0
1,6,0.8358,0.5710695,0.992406,0.9775525000000002,7,0
1,6,0.0031759999999999983,0.5674405,0.1387880000000002,0.9947314999999999,3,4
1,6,0.09906,0.3543735,0.24114000000000002,0.6454725,2,2
1,6,0.09906,0.3543735,0.24114000000000002,0.6454725,3,2
1,7,0.0033079999999999915,0.570406,0.15909,0.9930019999999998,2,4
1,7,0.0033079999999999915,0.570406,0.15909,0.9930019999999998,3,4
1,7,0.09841000000000001,0.3554695,0.23734,0.6533245,2,2
1,7,0.09841000000000001,0.3554695,0.23734,0.6533245,3,2
1,7,0.325579,0.29355600000000004,0.546621,0.8591660000000001,2,1
1,7,0.325579,0.29355600000000004,0.546621,0.8591660000000001,3,1
1,7,0.7428595,0.5725465,0.9827285,0.9861135000000001,5,0
1,8,0.004540500000000003,0.5759335,0.1675695,0.9903485,3,4
1,8,0.09621799999999998,0.36168700000000004,0.25073,0.655775,3,2
1,8,0.6350870000000001,0.6127644999999999,0.8517730000000001,0.9865354999999999,5,0
1,9,0.3288245,0.29604900000000006,0.5402815000000001,0.8635630000000001,2,1
```

7 修正ava_train_temp.csv

2024年8月27日　　9:53

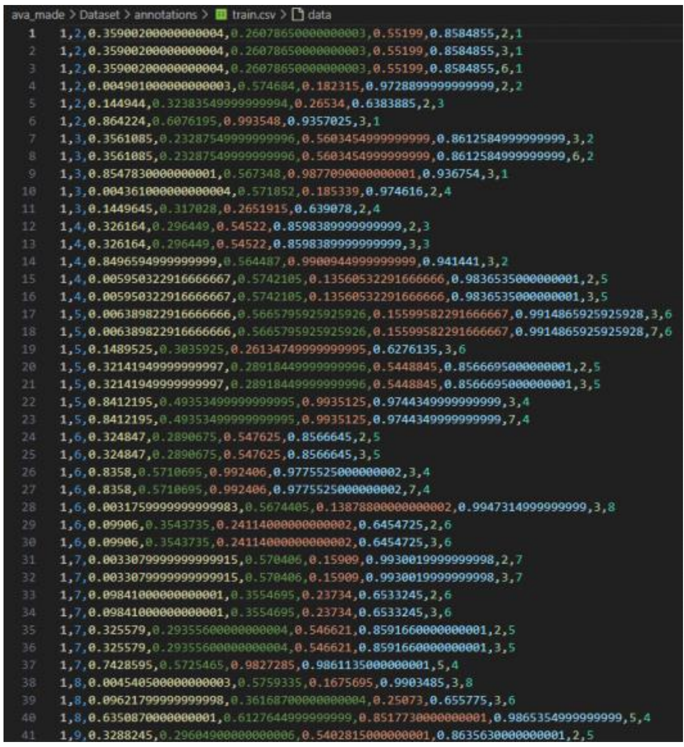
在上一步中融合后的文件中出现“-1”的数据，表示动作和ID两个框没有匹配，需要去除掉这些出现-1的数据条

结果在：Dataset/annotations/train.csv

```
import csv
train_temp_path = './train_temp.csv'
train_temp = []

with open(train_temp_path) as csvfile:
    csv_reader = csv.reader(csvfile)  # 使用csv.reader读取csvfile中的文件
    for row in csv_reader:
        train_temp.append(row)
def update_train_temp(videoName,index,maxId):
    for index2 in range(len(train_temp)):
        data = train_temp[index2]
        if index2 < index:
            continue
        if videoName == data[0]:
            if index2 == index:
                train_temp[index][-1] = maxId + 1
                # 并且查查ava_train_temp[index]后面10个的坐标是否与ava_train_temp[index]一致
                # 如果一致，就让该ava_train_temp[index + n]的ID与ava_train_temp[index]一致
                x1 = float(train_temp[index][2])
                y1 = float(train_temp[index][3])
                x2 = float(train_temp[index][4])
                y2 = float(train_temp[index][5])
                for index3 in range(10):
                    if train_temp[index+index3+1][-1] == '-1':
                        xT1 = float(train_temp[index+index3+1][2])
                        yT1 = float(train_temp[index+index3+1][3])
                        xT2 = float(train_temp[index+index3+1][4])
                        yT2 = float(train_temp[index+index3+1][5])
                        if abs(x1-xT1)<0.005 and abs(y1-yT1)<0.005 and abs(x2-xT2)<0.005 and abs(y2-yT2)<0.005:
                            train_temp[index+index3+1][-1] = maxId + 1
                        else:
                            break
                    else:
                        break

            else:
                if train_temp[index2][-1] == '-1':
                    continue

                xTT1 = float(train_temp[index2][2])
                yTT1 = float(train_temp[index2][3])
                xTT2 = float(train_temp[index2][4])
                yTT2 = float(train_temp[index2][5])
                if abs(x1-xTT1)<0.005 and abs(y1-yTT1)<0.005 and abs(x2-xTT2)<0.005 and abs(y2-yTT2)<0.005:
                    continue
                train_temp[index2][-1] = int(train_temp[index2][-1]) + 1
temp = train_temp
# dicts存放修正后的ava_train_temp
dicts = []
# personID_index 用来记录修正进行到的位置
#personID_index = 0
# maxId用来记录当前视频的进行中最大的ID
maxId = -1
# videoName 用来记录当前视频的名字
videoName = ''
for index in range(len(train_temp)):
    data = train_temp[index]
    # 判断是否切换视频，如果切换视频，
    # 那么videoName改变、maxId重制
    if videoName!=data[0]:
        videoName = data[0]
        maxId = -1

    if maxId < int(data[-1]):
        maxId = int(data[-1])
    if data[-1] == '-1':
        update_train_temp(videoName,index,maxId)
        # 经过 update_ava_train_temp 后，data[-1]为‘-1’对应的坐标的ID赋予maxID+1，那么最高值也要+1
        maxId = maxId + 1

with open('./annotations/train.csv',"w") as csvfile:
    writer = csv.writer(csvfile)
    writer.writerows(train_temp)
```

命令行

```
cd  ava_made/Dataset

python train.py
```



ava_made > Dataset > annotations > train.csv > data

```
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,2,1
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,3,1
1,2,0.35900200000000004,0.26078650000000003,0.55199,0.8584855,6,1
1,2,0.004901000000000003,0.574684,0.182315,0.9728899999999999,2,2
1,2,0.144944,0.32383549999999994,0.26534,0.6383885,2,3
1,2,0.864224,0.6076195,0.993548,0.9357025,3,1
1,3,0.3561085,0.23287549999999996,0.5603454999999999,0.8612584999999999,3,2
1,3,0.3561085,0.23287549999999996,0.5603454999999999,0.8612584999999999,6,2
1,3,0.8547830000000001,0.567348,0.9877090000000001,0.936754,3,1
1,3,0.004361000000000004,0.571852,0.185339,0.974616,2,4
1,3,0.1449645,0.317028,0.2651915,0.639078,2,4
1,4,0.326164,0.296449,0.54522,0.8598389999999999,2,3
1,4,0.326164,0.296449,0.54522,0.8598389999999999,3,3
1,4,0.8496594999999999,0.564487,0.9900944999999999,0.941441,3,2
1,4,0.005950322916666667,0.5742105,0.13560532291666666,0.9836535000000001,2,5
1,4,0.005950322916666667,0.5742105,0.13560532291666666,0.9836535000000001,3,5
1,5,0.006389822916666666,0.5665795925925926,0.15599582291666667,0.9914865925925928,3,6
1,5,0.006389822916666666,0.5665795925925926,0.15599582291666667,0.9914865925925928,7,6
1,5,0.1489525,0.3035925,0.26134749999999995,0.6276135,3,6
1,5,0.32141949999999997,0.28918449999999996,0.5448845,0.8566695000000001,2,5
1,5,0.32141949999999997,0.28918449999999996,0.5448845,0.8566695000000001,3,5
1,5,0.8412195,0.49353499999999995,0.9935125,0.9744349999999999,3,4
1,5,0.8412195,0.49353499999999995,0.9935125,0.9744349999999999,7,4
1,6,0.324847,0.2890675,0.547625,0.8566645,2,5
1,6,0.324847,0.2890675,0.547625,0.8566645,3,5
1,6,0.8358,0.5710695,0.992406,0.9775525000000002,3,4
1,6,0.8358,0.5710695,0.992406,0.9775525000000002,7,4
1,6,0.0031759999999999983,0.5674405,0.13878800000000002,0.9947314999999999,3,8
1,6,0.09906,0.3543735,0.24114000000000002,0.6454725,2,6
1,6,0.09906,0.3543735,0.24114000000000002,0.6454725,3,6
1,7,0.0033079999999999915,0.570406,0.15909,0.9930019999999998,2,7
1,7,0.0033079999999999915,0.570406,0.15909,0.9930019999999998,3,7
1,7,0.09841000000000001,0.3554695,0.23734,0.6533245,2,6
1,7,0.09841000000000001,0.3554695,0.23734,0.6533245,3,6
1,7,0.325579,0.29355600000000004,0.546621,0.8591660000000001,2,5
1,7,0.325579,0.29355600000000004,0.546621,0.8591660000000001,3,5
1,7,0.7428595,0.5725465,0.9827285,0.9861135000000001,5,4
1,8,0.004540500000000003,0.5759335,0.1675695,0.9903485,3,8
1,8,0.09621799999999998,0.36168700000000004,0.25073,0.655775,3,6
1,8,0.6350870000000001,0.6127644999999999,0.8517730000000001,0.9865354999999999,5,4
1,9,0.3288245,0.29604900000000006,0.5402815000000001,0.8635630000000001,2,5
```

8 其他标注文件的生成

2024年8月10日 18:23

命令行

8.1 train_excluded_timestamps.csv

由于视频中没有需要排除的视频帧，所以这里就创建空的avaMin_train_excluded_timestamps.csv文件。

```
cd ava_made/Dataset/annotations
touch train_excluded_timestamps.csv
```

8.2 included_timestamps.txt

```
cd ava_made/Dataset/annotations
touch included_timestamps.txt
```

然后在included_timestamps.txt 中写入检测视频的秒数(我的数据使用的一段46秒的视频，去掉开头和结尾的两秒)

```
02
03
04
05
06
07
08
…
43
44
```

8.3 action_list.pbtxt

```
cd ava_made/Dataset/annotations
touch action_list.pbtxt
```

此文件包含动作的种类，如下：

```
item {
  name: "talk"
  id: 1
}
item {
  name: "watch"
  id: 2
}
item {
  name: "stand"
  id: 3
}
item {
  name: "stoop"
  id: 4
}
item {
  name: "walk"
  id: 5
}
item {
  name: "take off hook"
  id: 6
}
item {
  name: "catch"
  id: 7
}
```

8.4 dense_proposals_train.pkl

这个文件在第3步已经生成了，将其复制放在ava_made/Dataset/annotations目录下即可

9 val文件的生成

2024年8月27日　10:40

和train文件的生成方法相同，这里不再细说，需要的val文件如下：放在ava_made/Dataset/annotations目录下

dense_proposals_val.pkl
val.csv
val_excluded_timestamps.csv

10 rawframes文件夹

2024年8月27日　　10:44

在取名上，裁剪的视频帧存在与训练不匹配的问题，所以需要对/ava_made/Dataset/frames中的图片进行名字修改

首先将ava_made/dataset/frames即frames文件夹复制一份并命名为rawframes

然后运行命令行

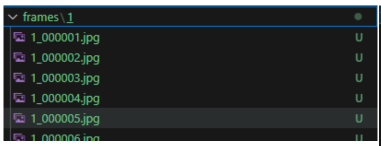
命令行

```
cd ava_made/Dataset/yolovDeepsort/mywork/
python change_raw_frames.py
```

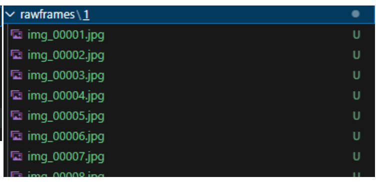
例如:

原本的名字：rawframes/1/1_000001.jpg

目标名字：rawframes/1/img_00001.jpg



```
frames \ 1
1_000001.jpg    U
1_000002.jpg    U
1_000003.jpg    U
1_000004.jpg    U
1_000005.jpg    U
```



```
rawframes \ 1
img_00001.jpg    U
img_00002.jpg    U
img_00003.jpg    U
img_00004.jpg    U
img_00005.jpg    U
img_00006.jpg    U
img_00007.jpg    U
```

```python
import os
for root, dirs, files in os.walk("../../Dataset/rawframes", topdown=False):
    for name in files:
        if 'checkpoint' in name:
            continue
        if "Store" in name:
            continue
        oldNamePath = os.path.join(root, name)

        tempName1 = name.split('_')[1] # 44_000054.jpg -> 000054.jpg
        tempName2 = tempName1.split('.')[0] # 000054.jpg -> 000054
        tempName3 = str(int(tempName2)).zfill(5) # 000054 -> 000054
        newName = 'img_' + tempName3 + '.jpg'
        newNamePath = os.path.join(root, newName)

        os.rename(oldNamePath,newNamePath)
```

11 标注文件修正

2024年8月27日 10:54

有部分的标注文件在字段类型上有些问题

所以需要修正

11.1 dense_proposals_train

```python
import pickle
import numpy as np
import csv
f = open('../../Dataset/annotations/dense_proposals_train.pkl','rb')
info = pickle.load(f, encoding='iso-8859-1')
dense_proposals_train = {}

for i in info:
    tempArr = np.array(info[i])
    dicts = []
    for index1,temp in enumerate(tempArr):
        temp = temp.astype(np.float64)
        for index2,x in enumerate(temp):
            if x < 0:
                temp[index2]=0.0
            if x > 1:
                temp[index2]=1.0
        dicts.append(temp)
    dense_proposals_train[i] = np.array(dicts)
# 保存为pkl文件
with open('../../Dataset/annotations/dense_proposals_train.pkl',"wb") as pklfile:
    pickle.dump(dense_proposals_train, pklfile)
```

11.2 dense_proposals_val

Val 和train的代码不同之处仅在于名称不同，这里不再展示

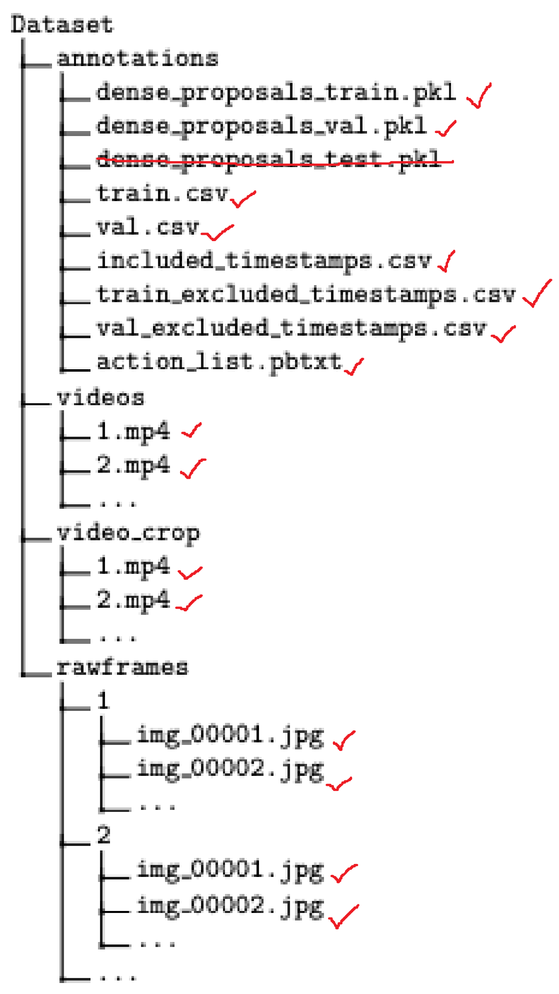
命令行

```
cd ava_made/yolovDeepsort/mywork
python change_dense_proposals_train.py
```

```
cd ava_made/yolovDeepsort/mywork
python change_dense_proposals_val.py
```

12 数据集文件总览

2024年8月27日 10:59



```
Dataset
├─annotations
│ ├─dense_proposals_train.pkl ✓
│ ├─dense_proposals_val.pkl ✓
│ ├─~~dense_proposals_test.pkl~~
│ ├─train.csv ✓
│ ├─val.csv ✓
│ ├─included_timestamps.csv ✓
│ ├─train_excluded_timestamps.csv ✓
│ ├─val_excluded_timestamps.csv ✓
│ └─action_list.pbtxt ✓
├─videos
│ ├─1.mp4 ✓
│ ├─2.mp4 ✓
│ └─...
├─video_crop
│ ├─1.mp4 ✓
│ ├─2.mp4 ✓
│ └─...
└─rawframes
  ├─1
  │ ├─img_00001.jpg ✓
  │ ├─img_00002.jpg ✓
  │ └─...
  ├─2
  │ ├─img_00001.jpg ✓
  │ ├─img_00002.jpg ✓
  │ └─...
  └─...
```

- Dataset
  - Annotations
    - dense_proposals_train.pkl
    - dense_proposals_val.pkl
    - train.csv
    - test.csv
    - included_timestamps.csv
    - train_excluded_timestamps.csv
    - val_excluded_timestamps.csv
    - action_list.pbtxt
  - videos
    - 1.mp4
    - 2.mp4
    - …
  - Video_corp
    - 1.mp4
    - 2.mp4
    - …
  - Rawframes
    - 1
      - Img_00001.jpg
      - Img_00002.jpg
      - …
    - 2
      - Img_00001.jpg
      - Img_00002.jpg
      - …
    - …